Fouille d'erreurs sur des sorties d'analyseurs syntaxiques

Benoît Sagot, Éric Villemonte de La Clergerie

Fouille d'erreurs sur des sorties d'analyseurs syntaxiques

Benoît Sagot – Éric de La Clergerie

INRIA Paris-Rocquencourt - Projet ALPAGE
Domaine de Voluceau
Rocquencourt, B.P. 105
F-78153 Le Chesnay cedex
{benoit.sagot,eric.de_la_clergerie}@inria.fr

RÉSUMÉ. *Nous présentons une méthode de fouille d'erreurs pour détecter automatiquement des erreurs dans les ressources utilisées par les systèmes d'analyse syntaxique. Nous avons mis en œuvre cette méthode sur le résultat de l'analyse de plusieurs millions de mots par deux systèmes d'analyse différents qui ont toutefois en commun le lexique syntaxique et la chaîne de traitement présyntaxique. Nous pouvons ainsi identifier des inexactitudes et des incomplétudes dans les ressources utilisées. En particulier, la comparaison des résultats obtenus sur les sorties des deux analyseurs sur un même corpus nous permet d'isoler les problèmes issus des ressources partagées de ceux issus des grammaires.*

ABSTRACT. *We introduce an error mining technique for automatically detecting errors in resources that are used in parsing systems. We applied this technique to parsing results produced on several million words by two distinct parsing systems, which share the syntactic lexicon and the pre-parsing processing chain. We are thus able to identify incorrectness and incompleteness sources in the resources. In particular, by comparing both systems' results, we are able to isolate problems coming from shared resources from those coming from grammars.*

MOTS-CLÉS : *analyse syntaxique, lexique syntaxique, fouille d'erreurs.*

KEYWORDS: *parsing, syntactic lexicon, error mining.*

TAL. Volume 49 – n° 1/2008, pages 41 à 60

1. Introduction

Le traitement automatique des langues est une tâche difficile, en partie à cause de la complexité et de la quantité des informations qui doivent être prises en compte à propos des mots et des constructions syntaxiques. Il est malgré tout nécessaire d'avoir accès à ces informations, qui sont rassemblées dans des ressources telles que les lexiques et les grammaires, et d'essayer d'y minimiser la quantité d'informations manquantes ou erronées. À cette fin, l'utilisation de ces ressources à grande échelle dans des systèmes d'analyse syntaxique est une approche très prometteuse (van Noord, 2004), et notamment en étudiant les situations qui conduisent à un échec de l'analyse : on peut apprendre de ses erreurs.

Dans cet article, nous présentons un modèle probabiliste qui permet d'identifier les formes et les bigrammes de formes, mais également les lemmes et les bigrammes de lemmes, qui posent problème, à partir du résultat d'un ou de plusieurs analyseurs sur un corpus. Afin de faciliter l'exploitation des résultats du modèle, et notamment pour identifier les causes des erreurs, nous avons développé une interface de visualisation. L'ensemble a été appliqué sur les résultats de deux analyseurs syntaxiques du français (FRMG et SXLFG-fr), qui utilisent un même lexique syntaxique (le Le*fff*) et une même chaîne de traitement présyntaxique (SXPipe), et sur différents types de corpus importants (plusieurs millions de mots). Enfin, diverses améliorations, extensions et adaptations des idées présentées ici sont mentionnées à la fin de l'article, montrant ainsi la richesse des idées sous-jacentes.

Bien que nous n'ayons mis en œuvre ces techniques que pour le français, elles sont fondamentalement indépendantes de la langue et du système d'analyse utilisé. Elles pourraient être appliquées telles quelles sur les résultats d'analyse produits par un système quelconque travaillant sur n'importe quelle langue. La seule information qui est exploitée est une valeur booléenne pour chaque phrase, qui indique si son analyse a été un succès ou un échec.

2. Principes

2.1. *Idée générale*

L'idée que nous avons mise en œuvre est la suivante, inspirée de (van Noord, 2004). Pour identifier les incomplétudes et les incorrections d'un système d'analyse syntaxique, on peut analyser un corpus de taille conséquente et étudier à l'aide d'outils statistiques ce qui différencie les phrases pour lesquelles l'analyse a réussi de celles pour lesquelles elle a échoué.

L'application la plus simple de cette idée consiste à chercher les formes, dites *suspectes*, qui se retrouvent fréquemment dans des phrases qui n'ont pu être analysées. C'est ce que fait (van Noord, 2004), sans toutefois chercher à identifier une forme suspecte pour chaque phrase non analysable, et donc sans prendre en compte le fait qu'il y a une cause d'erreur dans toute phrase non analysable. Il définit en effet le

taux de suspicion d'une forme f par le taux de phrases non analysables parmi celles contenant f.

Au contraire, nous allons chercher, pour chaque phrase dont l'analyse a échoué, la forme qui a le plus de chances d'être la cause de cet échec : c'est le *suspect principal* de la phrase. Il se peut que cette forme soit renseignée dans le lexique de façon incorrecte ou incomplète, qu'elle participe à des constructions non couvertes par la grammaire, ou qu'elle illustre des imperfections de la chaîne de traitement présyntaxique. Cette idée peut être facilement étendue aux suites de formes, et nous le ferons pour les bigrammes de formes, mais également aux lemmes (et aux suites de lemmes).

Cette idée peut être vue comme l'adaptation à un nouveau problème (et la simplification) de l'algorithme d'apprentissage automatique de lexiques morphologiques décrit dans (Sagot, 2005) : plutôt que de chercher à attribuer un lemme (forme canonique et table de flexion) à chaque forme, parmi plusieurs lemmes hypothétiques possibles (et générés automatiquement), on cherche ici à attribuer un suspect à chaque phrase, parmi plusieurs suspects hypothétiques possibles ; dans les deux cas, on itère un algorithme de point fixe qui fait un aller-retour entre le niveau local (la forme, la phrase) et le niveau global (l'ensemble des formes, l'ensemble des phrases). La tâche étudiée ici étant toutefois plus simple, l'algorithme utilisé s'en voit simplifié également.

2.2. *Modèle probabiliste*

On suppose donc que le corpus utilisé est découpé en phrases, elles-mêmes découpées en formes. On note p_i la i-ième phrase. On note $o_{i,j},\ (1 \leq j \leq |p_i|)$ les occurrences des formes constituant p_i, et on désigne par $F(o_{i,j})$ les formes correspondantes. Enfin, on note erreur la fonction qui à une phrase p_i associe 1 si l'analyse de p_i a échoué, et 0 sinon.

Soit $\mathcal{O}_f$ l'ensemble des occurrences d'une forme f dans le corpus :

$$\mathcal{O}_f = \{o_{i,j} | F(o_{i,j}) = f\}.$$

Le nombre d'occurrences de f dans le corpus est donc $|\mathcal{O}_f|$.

Définissons tout d'abord le *taux de suspicion moyen global* $\overline{S}$, qui est la probabilité moyenne qu'une occurrence donnée d'une forme soit la cause d'un échec d'analyse. Nous faisons l'hypothèse que l'échec de l'analyse d'une phrase est dû à une cause unique, et donc ici à une forme unique. Cette hypothèse, qui n'est pas nécessairement vérifiée, simplifie le modèle et donne des résultats pertinents. En notant $\mathrm{occ}_{\mathrm{total}}$ le nombre total de formes dans le corpus, on a donc :

$$\overline{S} = \frac{\Sigma_i \mathrm{erreur}(p_i)}{\mathrm{occ}_{\mathrm{total}}}.$$

Soit une forme f qui est la j-ième forme de la phrase p_i, c'est-à-dire que $F(o_{i,j}) = f$. Supposons que l'analyse de p_i a échoué : $\mathrm{erreur}(p_i) = 1$. On appelle *taux de suspi-*

cion de la j-ième forme $o_{i,j}$ de la phrase p_i la probabilité, notée $S_{i,j}$, que l'occurrence $o_{i,j}$ de la forme f soit responsable de l'échec de l'analyse de p_i. Si au contraire l'analyse de la phrase p_i a réussi, ses occurrences ont un taux de suspicion nul.

On définit alors le *taux de suspicion moyen* S_f de la forme f comme étant la moyenne des taux de suspicion de ses occurrences (toutes phrases confondues) :

$$S_f = \frac{1}{|\mathcal{O}_f|} \cdot \sum_{o_{i,j} \in \mathcal{O}_f} S_{i,j}.$$

Nous utilisons un algorithme de recherche de point fixe, en itérant un certain nombre de fois les calculs suivants. Supposons que l'on vient de terminer la n-ième itération : nous connaissons pour chaque phrase p_i et pour chaque occurrence $o_{i,j}$ de cette phrase l'estimation de son *taux de suspicion* $S_{i,j}$ par la n-ième itération, estimation notée $S_{i,j}^{(n)}$. On peut en déduire l'estimation de rang $n+1$ pour le taux de suspicion moyen de chaque forme f, noté $S_f^{(n+1)}$:

$$S_f^{(n+1)} = \frac{1}{|\mathcal{O}_f|} \cdot \sum_{o_{i,j} \in \mathcal{O}_f} S_{i,j}^{(n)}.$$

Ce taux[1] nous permet de calculer une nouvelle estimation des taux de suspicion des occurrences, en attribuant à chaque occurrence d'une phrase p_i un taux de suspicion $S_{i,j}^{(n+1)}$ égal à cette estimation $S_f^{(n+1)}$ du taux de suspicion moyen S_f de la forme correspondante, puis en normalisant à l'échelle de la phrase[2]. Ainsi :

$$S_{i,j}^{(n+1)} = \text{erreur}(p_i) \cdot \frac{S_{F(o_{i,j})}^{(n+1)}}{\sum_{1 \leq j \leq |p_i|} S_{F(o_{i,j})}^{(n+1)}}.$$

1. Nous avons également fait des expériences où S_f est estimé à l'aide d'un autre estimateur, le *taux de suspicion moyen lissé*, noté $\tilde{S}_f^{(n)}$, qui prend en compte le nombre d'occurrences de f. En effet, la confiance que l'on peut accorder à l'estimation $S_f^{(n)}$ est d'autant plus faible que le nombre d'occurrences occ_f est faible. D'où l'idée de lisser $S_f^{(n)}$ en le remplaçant par un barycentre $\tilde{S}_f^{(n)}$ de $S_f^{(n)}$ et de $\overline{S}$ dont le coefficient de pondération λ dépend de $|\mathcal{O}_f|$: si $|\mathcal{O}_f|$ est grand, $\tilde{S}_f^{(n)}$ sera proche de $S_f^{(n)}$; s'il est petit, il sera plus proche de $\overline{S}$:

$$\tilde{S}_f^{(n)} = \lambda(|\mathcal{O}_f|) \cdot S_f^{(n)} + (1 - \lambda(|\mathcal{O}_f|)) \cdot \overline{S}.$$

Dans ces expériences, nous avons utilisé la fonction de lissage $\lambda(|\mathcal{O}_f|) = 1 - e^{-\beta|\mathcal{O}_f|}$ avec $\beta = 0.1$. Mais ce modèle couplé au classement selon $M_f = S_f \cdot \ln|\mathcal{O}_f|$ (voir plus bas) donne des résultats similaires au modèle sans lissage. Nous présentons donc le modèle non lissé, qui a l'avantage de ne pas faire intervenir de fonction de lissage choisie empiriquement.

2. Les taux que nous calculons sont en effet des probabilités. Ainsi, $S_{i,j}^{(n)}$ peut s'interpréter comme la probabilité que l'échec de l'analyse de la phrase p_i soit causée par l'occurrence $o_{i,j}$.

À ce stade, la $n+1$-ième itération est terminée, et on peut recommencer à nouveau ces calculs, jusqu'à convergence sur un point fixe. L'amorçage de cet algorithme se fait en posant, pour une occurrence $o_{i,j}$ dans la phrase p_i, $S_{i,j}^{(0)} = \text{erreur}(p_i)/|p_i|$. Autrement dit, si l'analyse de p_i a échoué, on part d'une estimation où toutes ses occurrences ont une probabilité égale d'être la cause de l'échec.

Après quelques dizaines d'itérations de ce processus, on obtient donc des estimations du taux de suspicion moyen de chaque forme, ce qui permet :

– de repérer les formes les plus vraisemblablement responsables d'erreurs ;

– pour chaque forme f, d'identifier les phrases non analysables p_i où une de ses occurrences $o_{i,j} \in \mathcal{O}_f$ est un des principaux suspects et où $o_{i,j}$ a un taux de suspicion parmi les plus élevés parmi les occurrences de f.

L'algorithme a été implémenté en *perl*, en optimisant les structures de données pour réduire les coûts en mémoire et en temps. En particulier, les phrases stockent une liste d'occurrences qui pointent directement sur la structure associée à chaque forme.

On notera qu'une autre formalisation des idées mises en œuvre dans cet algorithme a été développée par François Yvon, en collaboration avec les auteurs de cet article. Cette formalisation, non encore publiée, fait usage d'un modèle de Markov caché à 3 états : chaque phrase est parcourue par ce modèle, le premier état étant celui où l'on n'a pas encore rencontré le suspect principal, le deuxième état correspondant à l'émission du suspect principal, et le dernier état étant celui où le suspect principal a déjà été émis. Il faut donc trouver pour chaque phrase la meilleure séquence d'états. Ainsi formalisé, le problème peut se ramener à une estimation du modèle, et l'aller-retour entre le niveau local et le niveau global devient un algorithme EM (*expectation-maximization*) standard, moyennant quelques petits changements dans le modèle de départ. Cette vision, qui permet une formalisation plus « standard » de l'algorithme, devrait donner lieu dans un avenir proche à une investigation plus poussée.

2.3. *Extensions du modèle*

Ce modèle donne déjà de très bons résultats, comme nous le verrons à la section 4. Cependant, on peut l'étendre de différentes façons, parmi lesquelles nous avons déjà implémenté certaines.

Tout d'abord, il est possible ne pas se restreindre aux formes. De fait, nous ne travaillons pas sur les formes seules, mais sur des couples formés d'une forme (une entrée du lexique) et du ou des tokens qui leur correspondent dans le texte brut[3].

3. Un token est une portion de texte délimitée par des espaces ou des tokens de ponctuation, alors qu'une forme est une unité linguistique syntaxiquement atomique. Ainsi, la chaîne de traitement présyntaxique SXPipe utilisée ici fait correspondre aux suites de tokens `aux`, `Le` (en début de phrase), `l'idée`, `à l'instar de` et `22` les suites de formes respectives *à le*, *le*,

Par ailleurs, on peut vouloir rechercher la cause de l'échec de l'analyse d'une phrase pas seulement dans la présence d'une forme dans cette phrase, mais aussi dans la présence d'un bigramme[4] voire d'un trigramme de formes. Il suffit pour cela de généraliser la notion de forme et d'occurrence, en considérant qu'une forme (généralisée) est un unigramme ou un bigramme de formes, et qu'une occurrence (généralisée) est une occurrence d'une forme généralisée, c'est-à-dire une occurrence d'un unigramme ou d'un bigramme de formes. Les résultats que nous présentons à la section 4 incluent cette extension, ainsi que la précédente.

L'autre généralisation possible consisterait à savoir prendre en compte des faits qui ne sont pas simultanés, mais qui sont des hypothèses concurrentes, qu'il faut par conséquent probabiliser aussi. Nous n'avons pas encore mis en œuvre un tel mécanisme. Il serait pourtant très intéressant, car il permettrait de dépasser le stade de la forme ou des n-grammes de formes et d'obtenir des généralisations, par exemple au niveau des lemmes.

3. Mise en œuvre

Nous avons appliqué ces principes pour rechercher les causes d'erreurs dans les sorties de deux systèmes d'analyse syntaxique profonde pour le français, FRMG et SXLFG-fr, que nous décrivons brièvement ci-dessous. À l'aide de ces deux systèmes, nous avons effectué l'analyse syntaxique de près d'un million de phrases du *Monde diplomatique*. Enfin, nous avons développé un environnement complet de visualisation des résultats de la fouille d'erreurs. La présente section décrit ces trois points successivement.

3.1. *Analyseurs*

Les deux systèmes d'analyse que nous avons utilisés reposent sur des analyseurs syntaxiques profonds non probabilistes. Ils ont en commun :

– le lexique syntaxique Le*fff* 2 (Sagot *et al.*, 2005 ; Sagot et Danlos, 2007), qui comporte 550 000 entrées (couvrant 400 000 formes distinctes) ; une entrée regroupe informations morphologiques, cadres de sous-catégorisation syntaxique (lorsque pertinent) et informations syntaxiques complémentaires, en particulier pour les formes verbales (contrôles, attributifs, impersonnels…) ;

– la chaîne de traitement présyntaxique SXPipe (Sagot et Boullier, 2008), qui convertit un texte brut en suite de treillis de mots présents dans le Le*fff* ; SXPipe

à_l'instar_de, *l' idée* et *_NUMBER* (le caractère espace représente une frontière entre formes). On pourra se reporter à (Sagot et Boullier, 2005) pour plus de précisions.

4. On pourrait généraliser cela à des n-grammes pour un n quelconque, mais plus n croît et plus le nombre d'occurrences des n-grammes diminue, ce qui conduit à des résultats statistiquement non significatifs.

comprend entre autres des modules de segmentation en phrases, de tokenization et correction orthographique, de détection d'entités nommées, et d'identification non déterministe des mots composés.

En revanche, FRMG et SXLFG-fr utilisent des analyseurs complètement différents, ne reposant ni sur le même formalisme, ni sur la même grammaire, ni sur le même générateur d'analyseurs. La comparaison des résultats de fouille d'erreurs sur les sorties de ces systèmes nous permet donc de distinguer les erreurs dues à L*efff* ou à SXPipe d'une part, et celles dues à l'une ou l'autre des grammaires d'autre part. Voyons plus en détail les caractéristiques de ces deux analyseurs.

L'analyseur FRMG (Thomasset et Villemonte de la Clergerie, 2005) s'appuie sur une grammaire compacte TAG du français générée à partir d'une métagrammaire. La compilation et l'exécution de l'analyseur se déroule dans le cadre du système DYALOG (Villemonte de la Clergerie, 2005).

L'analyseur SXLFG-fr (Boullier et Sagot, 2005b ; Boullier et Sagot, 2005a) est un analyseur LFG efficace et robuste. L'analyse est effectuée en deux phases. Tout d'abord, un analyseur à la Earley construit une forêt partagée représentant l'ensemble des analyses en constituants qui satisfont le squelette non contextuel de la grammaire. Puis les structures fonctionnelles sont construites de bas en haut, éventuellement en plusieurs passes. L'efficacité de l'analyseur repose sur diverses techniques de représentation compacte de l'information, sur l'emploi systématique de méthodes de partage de calculs et de structures, sur des techniques d'évaluation paresseuse, et sur l'élagage heuristique et quasiment non destructif en cours d'analyse.

Nos deux analyseurs implémentent aussi également des techniques évoluées de rattrapage et de tolérance d'erreurs, mais elles sont inutiles dans le cadre des travaux décrits ici, puisque nous ne cherchons qu'à distinguer les phrases qui ont une analyse complète (sans utilisation de techniques de rattrapage) de celles qui n'en ont pas.

3.2. *Corpus*

Nous avons analysé, à l'aide de ces deux systèmes, un corpus journalistique de plus de 960 000 phrases tirées d'un corpus d'articles du *Monde diplomatique* dont le style est naturellement exclusivement journalistique. Il s'agit d'un corpus brut dans le sens où aucun nettoyage n'a été effectué pour éliminer certaines séquences de caractères ne pouvant pas être considérées comme des phrases.

La table 1 donne quelques informations générales sur ce corpus et sur les résultats obtenus par les deux analyseurs, lors d'expériences menées à trois ans d'intervalle, nommées respectivement MD05 et MD08. Il est à noter que lors de la première expérience (en 2005), les deux analyseurs n'ont pas traité l'ensemble du corpus, n'avaient pas exactement le même jeu ni le même nombre de phrases, et n'utilisaient pas exactement la même notion de phrase. Lors de la deuxième expérience, à l'inverse, l'ensemble du corpus a été traité par les deux anayseurs, en partant de la même sortie

produite par SXPipe. Par ailleurs, entre les deux expériences, l'analyseur SXLFG-fr a changé quant à sa manière de considérer l'analyse d'une phrase comme correcte ou incorrecte, d'où la diminution apparente du taux de couverture.

corpus	#phrases	#succès (%)	#formes[5]	#occ	$\overline{S}$ (%)	Date
MD08/FRMG	961 945	538 674 (56,0 %)	406 008	20 669 843	2,04 %	07/08
MD08/SXLFG	961 945	469 283 (48,8 %)	406 008	20 669 843	2,38 %	10/08
MD05/FRMG	330 938	136 885 (41,3 %)	255 616	10 422 926	1,86 %	07/05
MD05/SXLFG	567 039	343 988 (60,7 %)	327 785	14 482 059	1,54 %	04/05

Tableau 1. *Informations générales sur le corpus et les résultats d'analyse*

3.3. *Environnement de visualisation des résultats*

Nous avons développé un outil de visualisation des résultats de la fouille d'erreurs, qui permet de les parcourir et de les annoter. Il s'agit d'une page Web qui fait usage de techniques de génération dynamique, en particulier *via javascript*. Un exemple est montré figure 1.

Pour cela, les suspects sont triés selon une mesure M_f modélisant pour une forme f l'intérêt qu'a l'utilisateur à tenter de corriger dans ses ressources l'erreur correspondante. Un utilisateur souhaitant privilégier les erreurs presque certaines sur les erreurs fréquentes peut visualiser les suspects en fonction de $M_f = S_f$. Un utilisateur souhaitant privilégier la fréquence d'une erreur potentielle au détriment de sa crédibilité peut les visualiser en fonction de[6] $M_f = S_f|\mathcal{O}_f|$. Une position intermédiaire, adoptée ici, consiste à les trier en fonction de $M_f = S_f \cdot \ln|\mathcal{O}_f|$.

L'environnement de visualisation permet de naviguer entre les suspects (triés) dans un menu à gauche de la page (**A**). Lorsque le token suspect est égal à la forme, seul le token est indiqué. Sinon, le token est séparé de la forme par « / ». Si l'on sélectionne un suspect, la partie droite de la page indique toutes sortes d'informations à son sujet. Après avoir indiqué son rang dans le classement selon la mesure M_f choisie (**B**), un champ est disponible pour ajouter ou éditer un commentaire associé au suspect (**D**). Ces commentaires, destinés au dépouillement des résultats par des linguistes ou par les développeurs des analyseurs et des grammaires, sont stockés dans une base

5. Cette colonne indique le nombre de couples token(s)/forme distincts, qui sont les suspects effectivement considérés (*cf.* 2.3), et non le nombre de formes distinctes trouvées en corpus.

6. Soit une forme f. Le taux de suspicion S_f peut être vu comme la probabilité qu'une occurrence particulière de f induise une erreur. Par conséquent, $S_f|\mathcal{O}_f|$ modélise le nombre d'occurrences de f en tant que source d'erreurs.

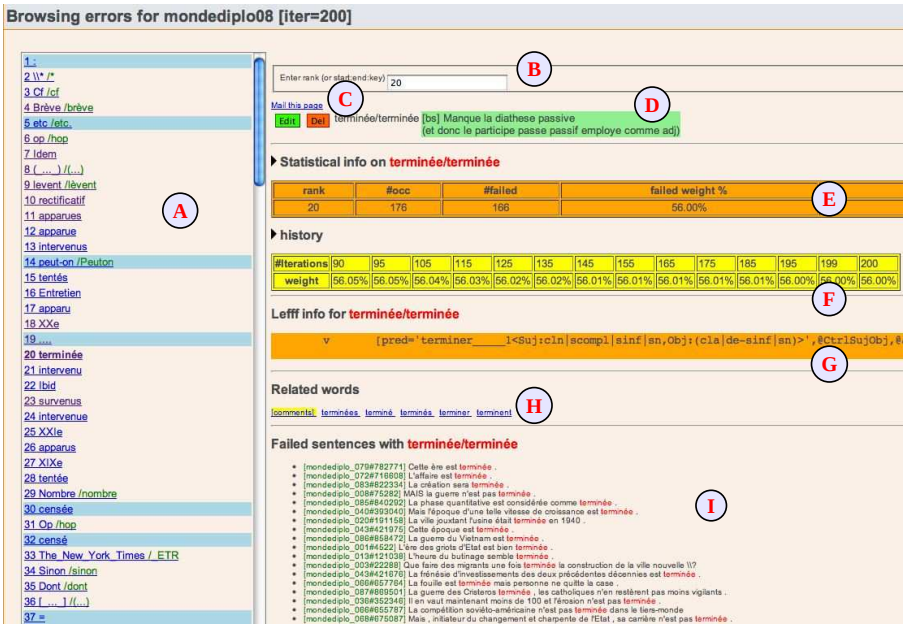

Figure 1. *Visualisation du résultat de la fouille d'erreurs (illustré pour MD08/FRMG)*

de données (SQLITE). Les suspects pour lesquels un commentaire a été rédigé sont mis en valeur par un fond bleu dans le menu de gauche. Des informations statistiques sont également fournies pour f (**E**), dont son nombre d'occurrences occ_f, le nombre d'occurrences de f appartenant à des phrases non analysables, l'estimation finale de son taux de suspicion moyen S_f, et le pourcentage $\mathrm{err}(f)$ de phrases non analysables parmi celles où f apparaît. Ces indications sont complétées par un bref historique (**F**) montrant la convergence de S_f. La partie inférieure de la page donne le moyen d'identifier les causes d'erreurs provenant de f en montrant (**G**) les entrées de f dans le lexique Le*fff*, en proposant des liens (**H**) vers des pages associés à des suspects proches (qui partagent un lemme commun avec f), et en listant (**I**) les phrases non analysables où f est le suspect principal et où une de ses occurrences a un taux de suspicion spécialement élevé[7].

7. De telles informations, qui sont très intéressantes pour les développeurs des ressources, ne peuvent pas être obtenues par des approches globales (qui se placent au niveau des formes et non au niveau de leurs occurrences), telles que l'approche de (van Noord, 2004), qui repose sur $\mathrm{err}(f)$. En effet, énumérer les phrases qui comportent une forme f donnée et qui n'ont pas reçu une analyse complète n'est pas suffisamment précis : cela montrerait à la fois des phrases où f est effectivement la cause de l'échec (par exemple parce qu'il manque un certain cadre

La page avec commentaires peut être envoyée par mail, par exemple au gestionnaire de Le*fff* ou au développeur de tel ou tel analyseur (**C**).

4. Résultats

Pour MD08/FRMG, par exemple, l'algorithme a effectué 200 itérations en environ une heure sur un PC à 3,2 GHz avec 1 Go de mémoire vive. Il a proposé 18 119 formes dites *pertinentes* (sur les 406 008 possibles), une forme pertinente étant une forme f qui vérifie les contraintes (arbitraires) de seuil suivantes : $S_f^{(200)} > 1,5 \cdot \overline{S}$ et $|\mathcal{O}_f| > 5$.

Nous ne pouvons pas encore assurer la convergence théorique de l'algorithme. Mais sur les 1 000 premières formes retournées pour MD08/FRMG, la dernière itération fait varier le taux de suspicion de 0,014 % en moyenne (le maximum est de 0,06 %, et seules 12 formes varient de plus de 0,01 %). En pratique, il n'est pas nécessaire de réaliser 200 itérations puisqu'on obtient déjà de bons résultats avec une cinquantaine d'itérations seulement.

La table 2 donne une idée de la répartition des suspects par rapport à leur fréquence, montrant que les formes rares ont proportionnellement plus de chances d'être suspectes que les formes de fréquence moyenne (les données sur les formes très fréquentes n'étant pas véritablement significatifs, en raison du faible nombre de formes concernées).

#occ	$>$ 100 000	$>$ 10 000	$>$ 1000	$>$ 100	$>$ 10
# formes	25	130	1 674	11 735	45 648
dont suspects	6	17	151	1733	10 936
	(24,0 %)	(13,1 %)	(9,0 %)	(14,8 %)	(24,0 %)

Tableau 2. *Répartition des suspects pour MD08/FRMG*

4.1. *Analyse des résultats*

La tables 3 montre des extraits des résultats obtenus en appliquant notre méthode sur les résultats d'analyse du corpus MD par FRMG. Elle donne les 20 couples token(s)/forme les mieux classés ainsi que ceux classés 50 à 55, pour montrer que les résultats restent pertinents au-delà des premiers rangs. Outre le taux de suspicion et le nombre d'occurrences, nous donnons le taux $\mathrm{err}(f)$ (utilisé par (van Noord, 2004)) qui est le pourcentage d'échecs d'analyse parmi les phrases comportant une occurrence de la forme f, ainsi qu'une analyse manuelle de la cause de l'erreur identifiée automatiquement.

---

de sous-catégorisation à son entrée lexicale), mais également des phrases dont l'analyse échoue pour une autre raison, totalement indépendante de f.

Rang	Token(s)/forme	$S_f^{(200)}$	$\lvert\mathcal{O}_f\rvert$	$\mathrm{err}(f)$	M_f	Origine de l'erreur
1	:	78 %	160997	98 %	9,36	SxPipe et/ou FRMG : problème de segmentation sur les « : ». Beaucoup d'exemples ou les 2 morceaux avant et apres les deux-points sont necessaires pour obtenir une phrase.
2	*	90 %	4200	100 %	7,52	corpus : pseudo-phrases comportant le seul mot « * »
3	Cf/cf	77 %	2996	100 %	6,13	SxPipe : bug
4	Brève/brève	81 %	1206	99 %	5,76	corpus : pseudo-phrases particulières
5	etc/etc.	59 %	955	100 %	4,07	Lefff : *etc.* n'est pas (qu')une ponctuation…
6	op/hop	61 %	604	99 %	3,90	SxPipe : bug pour *op. cit.*
7	Idem	71 %	168	100 %	3,64	corpus et FRMG : phrases de type *(1) Idem.*
8	(_..._)/(...)	43 %	3267	85 %	3,46	FRMG : gestion du *(...)*
9	levent/lèvent	95 %	34	100 %	3,37	corpus : noms d'auteurs d'articles en minuscules
10	rectificatif	64 %	186	98 %	3,36	corpus : pseudo-phrases particulières
11	apparues	70 %	81	100 %	3,07	Lefff : manque comme adjectif
12	apparue	62 %	138	100 %	3,05	Lefff : idem
13	intervenus	63 %	120	98 %	3,03	Lefff : idem
14	peut-on/Peuton	52 %	346	94 %	3,02	SxPipe : bug
15	tentés	61 %	145	99 %	3,01	Lefff : manque comme adjectif
16	Entretien	54 %	268	100 %	3,01	grammaire : pseudo-phrases du type *(1) Entretien avec…*
17	apparu	56 %	204	100 %	2,99	Lefff : manque comme adjectif
18	XXe	49 %	434	93 %	2,95	SxPipe : bug sur les ordinaux romains
19	....	64 %	99	100 %	2,93	FRMG : ajout de .... comme ponctuation finale
20	terminée	56 %	176	94 %	2,90	Lefff : manque comme participe passé passif
50	azimuts	52 %	111	87 %	2,45	Lefff : manque « tous azimuts » comme adverbe (et adjectif)
51	The_Washington_Post/_ETR	61 %	52	98 %	2,42	traitement des références bibliographiques
52	The_Guardian/_ETR	51 %	114	92 %	2,41	Idem
53	elle-même	33 %	1400	75 %	2,40	FRMG : manque les constructions de type *Pierre dit lui-même…* et de type *… le système lui-même*
54	De_sorte_que/de_sorte_que	60 %	53	98 %	2,40	FRMG : phrases de type *De sorte que P.*
55	lui-même	31 %	2504	76 %	2,40	FRMG : comme pour *elle-même*

Tableau 3. *Analyse des 20 premières formes et de celles de rang 50 à 55 pour MD08/FRMG (classées selon* $M_f = S_f \cdot \ln|\mathcal{O}_f|$*)*

On constate que les résultats sont satisfaisants : les deux tableaux identifient correctement des erreurs, réparties en quatre sources : les erreurs dans le lexique Le*fff*, les erreurs dans la chaîne de traitement présyntaxique SXPipe, les imperfections de la grammaire, mais aussi les problèmes issus du corpus en raison du fait qu'il s'agit d'un corpus brut.

Nous avons également appliqué l'algorithme à un corpus plus petit (moins de 40 000 phrases) et plus varié, le corpus développé dans le cadre de la campagne EASy d'évaluation des analyseurs syntaxiques pour le français. Les résultats sont également pertinents, mais parfois plus délicats à interpréter, suite à la taille limitée du corpus, à sa grande hétérogénéité et à la présence de sous-corpus de courriers électroniques et de transcription de corpus oraux qui introduisent beaucoup de bruit. L'importance des erreurs de segmentation (dues à la fois à SXPipe et au corpus lui-même, déjà segmenté) s'en trouve renforcée. Ceci montre que l'algorithme présenté ici ne fonctionne pleinement que sur un corpus suffisamment volumineux ou suffisamment homogène pour que des statistiques pertinentes puissent être effectuées sur les résultats d'analyse.

4.2. *Comparaison avec d'autres algorithmes*

Afin de valider notre approche, nous avons comparé nos résultats avec ceux produits par deux autres algorithmes pertinents :

– l'algorithme de van Noord (van Noord, 2004) (au niveau des formes, non itératif), qui repose sur $\mathrm{err}(f)$ (le taux de phrases non analysables parmi les phrases contenant la forme f) ;

– un algorithme standard de maximisation de l'entropie (au niveau des occurrences des formes, itératif) qui évalue la contribution de chaque forme au succès ou à l'échec d'une phrase (nous avons utilisé pour cela le paquetage MEGAM (Daumé III, 2004)).

Ainsi que nous l'avons fait pour notre propre algorithme, nous ne classons pas directement les formes en fonction de leur taux de suspicion S_f calculé par ces algorithmes. Nous utilisons en effet la mesure M_f présentée ci-dessus ($M_f = S_f \cdot \ln |\mathcal{O}_f|$). L'utilisation directe de la mesure de van Noord produit des suspects qui sont des mots très rares, ce qui montre l'importance d'un bon équilibre entre taux de suspicion et fréquence (comme remarqué par (van Noord, 2004) dans la discussion de ses résultats). Cette remarque s'applique également à la mesure obtenue par maximisation de l'entropie.

La table 4 montre pour les trois algorithmes les 10 formes suspectes les mieux classées, assorties d'une évaluation manuelle de leur pertinence. On voit immédiatement que notre approche conduit aux meilleurs résultats. La technique de van Noord a été développée au départ pour trouver les erreurs de ressources ayant déjà une couverture très importante. Dans nos systèmes, dont le développement est moins avancé, cette technique classe comme formes les plus suspectes celles qui sont tout simplement les plus fréquentes. Il semble que ce soit le cas également pour l'algorithme reposant sur la maximisation de l'entropie, ce qui montre qu'il est important de prendre en compte

le fait qu'il y a au moins une cause d'erreur par phrase dont l'analyse a échoué, pas seulement dans le but d'obtenir un suspect principal pour chaque phrase en échec, mais également pour obtenir des résultats globaux satisfaisants.

	Cet article		Global		Maxent	
Rang	Token(s)/forme	Éval	Token(s)/forme	Éval	Token(s)/forme	Éval
1	...	++	...	+	...	++
2	Cf	++	d'œuvre/œuvre	-	Cf	++
3	Cf/cf	++	d'œuvre/d'	-	Cf/cf	++
4	Traduit/traduit	++	,	-	être	-
5	Chacun/chacun	++	que	-	d'œuvre/d'	-
6	Prenons/prenons	++	plus	-	d'œuvre/œuvre	-
7	Où/où	++	et	-	lui	-
8	Seule/seule	++	les	-	sont	-
9	../...	+	de	-	faire	-
10	mort	++	Titre/titre	-	était	-

Tableau 4. *Les 10 formes suspectes les mieux classées selon la mesure M_f, telle qu'elle est calculée par différents algorithmes : le nôtre (*cet article*), le taux* $\mathrm{err}(f)$ *de van Noord (*global*) et un algorithme standard de maximisation de l'entropie (*maxent*). Ces résultats sont ceux de MD08/*SXLFG*-fr*

4.3. *Comparaison entre les deux analyseurs*

Nous avons complété l'étude des résultats séparés de nos deux systèmes d'analyse par un couplage des deux résultats. Pour ce faire, nous avons calculé pour chaque forme la moyenne harmonique des mesures $M_f = S_f \cdot \ln |\mathcal{O}_f|$ obtenues pour chaque système d'analyse. Les résultats sont intéressants, car ils identifient des erreurs provenant en majorité des ressources partagées entre les deux systèmes (le lexique L*efff* et la chaîne de traitement présyntaxique SXPipe). Bien que certaines erreurs soient issues d'incomplétudes communes aux deux grammaires, c'est néanmoins un moyen efficace d'obtenir une première répartition entre sources d'erreurs différentes.

À titre d'illustration, la table 5 montre quels sont les 15 couples token(s)/forme dont la moyenne harmonique des taux de suspicion obtenus par FRMG et SXLFG-fr est la plus grande (sur les résultats d'analyse du corpus MD).

4.4. *Passage aux bigrammes de formes*

Comme indiqué précédemment, nous avons également effectué des expériences au cours desquelles nous considérons non seulement les formes mais également les bigrammes de formes comme causes potentielles de l'échec de l'analyse. Cette approche permet d'identifier des situations où une forme n'est pas en soi la cause pertinente de

Rang	Token(s)/forme	$\bar{M}_f$	Origine de l'erreur
1	Cf/cf	2,2	
2	...	1,3	
3	Là/là	1,2	grammaires : phrases à sujet inversé mal traitées
4	Restent/restent	1,1	grammaires : idem
5	XXIe	1,1	SXPipe : bug d'identification des siècles
6	XXe	1,1	SXPipe : idem
7	Andes/Ande	1,0	Le*fff* : Ajouter les principaux massifs montagneux
8	Vient/vient	1,0	grammaires : comme pour *Restent/restent*
9	survenus	1,0	Le*fff* : Manque comme adjectif
10	Sont/sont	1,0	Le*fff* et grammaires : Constr. 'prep+adv' pour certains adv
11	autocentré	0,9	Le*fff* : Manque comme adjectif
12	Viennent/viennent	0,9	grammaires : comme pour *Restent/restent*
13	Quiconque/quiconque	0,9	grammaires : constructions de type *quiconque fait ceci fait cela* mal gérées
14	intervenus	0,9	Le*fff* : Manque comme adjectif
15	Etant/étant	0,9	Le*fff* et/ou grammaires : *étant donné(e)(s)*

Tableau 5. *Couples token(s)/forme maximisant la moyenne harmonique $\bar{M}_f$ des mesures M_f^{FRMG} et $M_f^{\text{SXLFG-fr}}$ obtenues par FRMG et* SXLFG*-fr*

Rang	Tokens/formes	M_f	Origine de l'erreur
3	Ce/ce qui	2,7	grammaire : phrases nominales débutant par *Ce qui* pas traitées
4	Où/où va	2,4	grammaire : certaines interrogatives directes sont mal traitées
9	fait partie	2,1	grammaire : construction *faire partie de* mal gérée
11	auteur notamment	2,1	grammaire : *auteur notamment de* : adverbe particulier à une place particulière
13	la mort	2,0	Le*fff* : *mort* est absent comme substantif féminin

Tableau 6. *Bigrammes de formes les mieux classés pour MD08/*SXLFG*-fr (les formes classées entre ces bigrammes ne sont pas montrées ; le classement est effectué selon la mesure $M_f = S_f \cdot \ln |\mathcal{O}_f|$)*

l'erreur, mais où elle conduit souvent à un échec de l'analyse lorsqu'elle est immédiatement précédée ou suivie d'une autre forme particulière.

La table 6 montre les bigrammes de formes les mieux classés (les formes qui sont classées entre ces bigrammes ne sont pas montrées, pour mettre en avant les résultats sur les bigrammes), avec les mêmes données qu'à la table 3.

5. Développements récents

En s'appuyant sur le modèle et sur les résultats présentés ci-dessus, un certain nombre d'expériences récentes et en cours cherchent à tirer le meilleur parti de l'analyse des causes d'erreurs dans les analyseurs syntaxiques. Nous allons décrire très rapidement trois de ces expériences, qui utilisent les idées décrites ici avec différents objectifs :

– aider à l'adaptation du système d'analyse FRMG et du lexique Le*fff* pour un corpus spécialisé (corpus botanique) à l'aide de la technique de fouille d'erreurs telle qu'elle est décrite dans (Role *et al.*, 2007) ;

– utiliser l'algorithme d'identification des suspects pour associer non pas un suspect unique à chaque phrase inanalysable, mais un gouverneur unique à chaque forme d'une phrase : on obtient ainsi un désambiguïsateur statistique appris de façon endogène ; cette idée, particulièrement utile dans le cas de corpus avec de nombreux mots inconnus, a été mise en œuvre là aussi sur un corpus botanique (Fernandez *et al.*, 2007a ; Role *et al.*, 2007 ; Fernandez *et al.*, 2007b) ;

– au-delà de l'identification d'entrées lexicales suspectes, proposer automatiquement au développeur de ressources des corrections pertinentes (Nicolas *et al.*, 2007a ; Nicolas *et al.*, 2007b).

5.1. *Améliorer la couverture d'un analyseur syntaxique sur un corpus spécialisé à l'aide de la fouille d'erreurs*

Dans le cadre du projet Biotim[8], un important corpus spécialisé dans le domaine de la botanique a été analysé à l'aide du système d'analyse FRMG, précédé de la chaîne de traitement présyntaxique SXPipe. Ce corpus est une flore du Sénégal, composée d'une quarantaine de volumes publiés entre 1963 et 2001. Malgré cet étalement dans le temps, la structure des documents reste relativement constante. En particulier, chaque espèce est décrite selon une certaine structure, y compris une partie descriptive, qui énumère des caractères distinctifs de l'espèce, sous forme de phrases nominales juxtaposées. On peut noter une grande quantité d'adjectifs (dénotant essentiellement des formes, couleurs et textures) ainsi que d'adverbes de fréquence et d'intensité (*légèrement*, *particulièrement*). De nombreuses entités nommées dénotant des dimensions sont également présentes, ainsi que des fautes typographiques résultant d'erreurs de reconnaissance de caractères[9].

8. L'objectif du projet Biotim était de concevoir des méthodes génériques d'analyse automatique de masses de données regroupant textes et images pour acquérir une surcouche sémantique commune et, à partir de ce premier résultat, développer des méthodes génériques d'interrogation plurimodale des données ainsi structurées.

9. Le corpus est en effet issu d'un processus de numérisation puis de reconnaissance automatique des caractères (OCR).

Sur des corpus généraux, FRMG atteint des taux de couvertures de 40 % à 50 % pour des analyses complètes, comme nous l'avons vu précédemment. Sur les corpus botaniques, le premier passage a fourni une maigre couverture de 36 %, dus à de nombreux problèmes liés au traitement présyntaxique (mauvaise structuration logique, contenu résiduel non textuel, mauvaise détection d'entités nommées, mauvaise segmentation, ...), à des entrées lexicales incomplètes et, bien sûr, à des constructions syntaxiques manquantes.

Pour améliorer la couverture de la chaîne de traitement, les techniques de fouille d'erreurs précédemment décrites ont été utilisées (Role *et al.*, 2007). En particulier, la fouille d'erreurs a été très utile pour identifier de nombreux mots techniques inconnus de Le*fff* et que SXPipe corrige en mots connus proches. Ce mécanisme d'autocorrection est très utile dans le cadre de corpus à faible fréquence de mots inconnus car il permet de traiter les inévitables coquilles. Mais, dans le cas présent, il a d'abord commencé à générer des fausses corrections qui ont ensuite provoqué des erreurs pour l'analyse. Ces erreurs ont été repérées par la fouille d'erreurs, et les mots inconnus correctement intégrés. Progressivement, le mécanisme d'auto-correction devient alors intéressant pour corriger les nombreuses coquilles issues de la phase de numérisation et non encore corrigées.

La fouille d'erreurs s'est aussi révélée utile pour détecter les termes techniques existant également comme termes généraux dans le Le*fff* mais avec des informations lexicales incompatibles. Ainsi, le terme *trident* est utilisé comme adjectif dans le corpus mais n'est enregistré que comme nom commun dans le Le*fff*.

En analysant quatorze fois le corpus (environ 80 000 phrases, selon le niveau de filtrage sur les phrases) et en exploitant le retour fourni à chaque passage par la fouille d'erreurs, la couverture de l'analyseur est passée de 36 % à 67 %. La raison première expliquant cette importante progression est la relative homogénéité de ce corpus technique, reposant sur un style assez strict et un vocabulaire certes technique mais réduit et à haute fréquence d'occurrences. Toute erreur systématique a alors tendance à se manifester fréquemment et se détecte donc aisément. Bien que les performances soient déjà bonnes, il est probablement encore possible de les améliorer au travers de quelques tours supplémentaires, mais, bien sûr, les rendements sont décroissants. Ces résultats montrent toutefois à quel point la technique de fouille d'erreurs décrite dans ce chapitre peut être utile pour adapter une (méta-)grammaire et un lexique généraux à un corpus spécialisé, avec ses termes et ses constructions spécifiques.

5.2. *Utiliser l'algorithme de fouille d'erreurs pour apprendre un désambiguïsateur endogène*

Les travaux préliminaires décrits dans (Fernandez *et al.*, 2007a ; Role *et al.*, 2007 ; Fernandez *et al.*, 2007b) tentent de décliner l'algorithme exposé à la section 2 dans un contexte différent. Nous avons vu que cet algorithme est lui-même adapté de celui présenté (Sagot, 2005), où il était appliqué pour l'apprentissage automatique de

lexiques morphologiques. Ici, l'algorithme est utilisé pour aider à la désambiguïsation des analyses syntaxiques, et notamment pour les termes et les mots inconnus.

Ce travail, effectué là aussi sur les corpus botaniques du projet Biotim, a appliqué l'algorithme avec deux objectifs distincts :

– attribuer à une forme donnée (inconnue, ou intrinsèquement ambiguë), sa catégorie la plus probable au vu de l'ensemble de ses occurrences dans le corpus ;

– identifier le gouverneur d'une forme donnée, là aussi en alternant une analyse au niveau de la phrase et une normalisation à l'échelle de tout le corpus.

Si les résultats obtenus sont bons, en particulier pour la détermination de la catégorie, c'est en partie à cause du caractère technique du corpus, dont on constate qu'il tend à n'attribuer qu'une seule catégorie aux termes, reflétant leur fréquente monosémie. Il en va de même, quoique dans une moindre mesure, pour la désambiguïsation des relations de dépendance.

Une fois les catégories et les relations de dépendances ainsi obtenues, l'idée est de s'en servir comme point de départ pour l'induction automatique des classes sémantiques, en identifiant par exemple les termes qui ont des dépendants comparables. L'objectif final, conformément à l'ambition du projet Biotim, est la constitution automatique d'une ontologie du domaine botanique à partir de l'analyse des corpus.

5.3. *Proposer des suggestions de corrections*

L'interface de visualisation des suspects et des phrases associés, que nous avons présentée à la section 3, est déjà d'une grande aide pour repérer des corrections à apporter au Le*fff* et aux analyseurs. Néanmoins, il est possible d'aller plus loin en essayant de proposer automatiquement des hypothèses de corrections pour les entrées lexicales des suspects.

Ce travail en cours, qui a déjà donné lieu à plusieurs publications (Nicolas *et al.*, 2007a ; Nicolas *et al.*, 2008), s'appuie sur la réanalyse des phrases associées à un suspect S en remplaçant les occurrences de celui-ci par un *joker*, c'est-à-dire une forme spéciale aux informations lexicales très sous-spécifiées.

Un premier résultat intéressant, qui confirme quantitativement la validité de l'algorithme de fouille d'erreurs, est le suivant : le taux d'analyse augmente d'autant plus que les suspects remplacés par un joker avaient un taux de suspicion plus élevé.

L'objectif principal est cependant d'examiner quelles informations lexicales se voient attribuer les jokers au sein des résultats de la réanalyse. On voit en effet émerger, sur l'ensemble des occurrences d'une forme ou d'un lemme suspect donné, des tendances sur les rôles joués par ces jokers dans les analyses. Par exemple, si le mot *forme* manque dans le lexique en tant que nom commun, et se retrouve donc suspect dans un certain nombre de phrases non analysables, on voit apparaître de nombreuses phrases dans lesquels le joker qui a remplacé *forme* est utilisé en tant que nom commun. Ce

type de tendances permettent de suggérer des corrections. Bien que prometteurs, les résultats sont encore fragmentaires et soulignent la difficulté de différencier certaines constructions syntaxiques ayant tendance à apparaître dans des contextes similaires.

6. Conclusions et perspectives

L'analyse de corpus importants permet donc de mettre en place des techniques de fouille d'erreurs, pour identifier les incomplétudes et les inexactitudes dans les différentes ressources utilisées par un système d'analyse syntaxique complet. La technique décrite ici et mise en œuvre sur les formes et les bigrammes de formes (ou de couples token(s)/forme) nous a déjà permis de détecter nombre d'erreurs et de manques dans notre lexique Le*fff*, de révéler des comportements inappropriés dans notre chaîne de traitement présyntaxique SXPipe, et de mettre en avant le manque de couverture de nos grammaires pour certains phénomènes.

Entre les expériences MD05 et MD08, il est difficile d'identifier la contribution relative des travaux décrits dans cet article et des autres améliorations apportées aux composants de la chaîne de traitement linguistique dans l'améliorations de ses performances (par exemple dans l'augmentation du taux de couverture de FRMG). Toutefois, d'un point de vue pratique, nous avons grandement bénéficié de cette technologie de fouille d'erreurs, en particulier pour l'amélioration du Le*fff* et la correction de problèmes dans SXPipe. La diminution globale des taux de suspicion entre MD05 et MD08 pourrait en être la trace.

Nous souhaitons mettre en place différentes améliorations et extensions de ce travail. Tout d'abord, l'interface est améliorable, de même que l'implémentation de l'algorithme itératif. En particulier, les travaux en cours en collaboration avec François Yvon sur la description du modèle sous forme de modèle de Markov caché pourrait permettre, *via* un algorithme de type EM, une meilleure description de l'algorithme utilisé.

Ensuite, nous pensons intégrer au modèle la possibilité que les faits pris en compte (ici les occurrences) ne soient pas nécessairement tous certains, mais puissent être parfois en concurrence les uns avec les autres. Par exemple, pour une forme donnée, plusieurs lemmes sont souvent possibles. Les probabiliser permettrait donc de rechercher les *lemmes* les plus suspects.

Enfin, nous comptons poursuivre les travaux préliminaires sur la construction automatiques d'hypothèses de corrections pour les entrées lexicales correspondant aux formes les plus suspectes.

Le travail présenté ici participe d'une approche cohérente du développement de ressources, où des outils automatiques produisent des informations aussi pertinentes que possibles, pour préparer le travail manuel d'extension, de correction et de validation. En effet, les outils d'acquisition automatique d'informations linguistiques permettent d'accélérer considérablement le développement de ressources, mais ils ne

sont pas là pour remplacer le travail du linguiste. L'impératif de qualité, de précision et de pertinence linguistique fait de la validation manuelle une étape incontournable, dont les outils automatiques cherchent à minimiser fortement le coût.

7. Bibliographie

Boullier P., Sagot B., « Analyse syntaxique profonde à grande échelle : SXLFG », *Traitement Automatique des Langues (T.A.L.)*, 2005a.

Boullier P., Sagot B., « Efficient and robust LFG parsing : SxLfg », *Proceedings of IWPT'05*, Vancouver, Canada, October, 2005b.

Daumé III H., « Notes on CG and LM-BFGS Optimization of Logistic Regression », 2004, Paper available at `http://www.isi.edu/~hdaume/docs/daume04cg-bfgs.ps`, implementation available at `http://www.isi.edu/~hdaume/megam/`.

Fernandez M., Villemonte de La Clergerie E., Vilares M., « From text to knowledge », *Proc. of EUROCAST'07 (Eleven international conference on Computer Aided Systems theory)*, 2007a.

Fernandez M., Villemonte de La Clergerie E., Vilares M., « Knowledge Acquisition through Error-Mining », *Proc. of International Conference RANLP'07*, Borovets, Bulgaria, September, 2007b.

Nicolas L., Farré J., Villemonte de la Clergerie E., « Confondre le coupable : Corrections d'un lexique suggérées par une grammaire », *Proc. of TALN'07*, 2007a.

Nicolas L., Farré J., Villemonte de la Clergerie E., « Mining Parsing Results for Lexical Corrections », *Proc. of the 3rd Language & Technology Conference (LTC)*, Poznań, Poland, October, 2007b.

Nicolas L., Sagot B., Villemonte de La Clergerie E., Farré J., « Computer aided correction and extension of a syntactic wide-coverage lexicon », *Actes de CoLing 2008*, Manchester, Royaume-Uni, 2008.

Role F., Gavilanes M. F., Villemonte de la Clergerie E., « Large-scale Knowledge acquisition from botanical texts », *Proc. of NLDB'07*, 2007.

Sagot B., « Automatic acquisition of a Slovak lexicon from a raw corpus », *Lecture Notes in Artificial Intelligence 3658 (© Springer-Verlag), Proceedings of TSD'05*, Karlovy Vary, Czech Republic, p. 156-163, September, 2005.

Sagot B., Boullier P., « From raw corpus to word lattices : robust pre-parsing processing », *Proceedings of L&TC 2005*, Poznań, Pologne, 2005.

Sagot B., Boullier P., « SXPipe 2 : architecture pour le traitement présyntaxique de corpus bruts », *Traitement Automatique des Langues (T.A.L.)*, 2008. à paraître.

Sagot B., Clément L., Villemonte de la Clergerie E., Boullier P., « Vers un méta-lexique pour le français : architecture, acquisition, utilisation », March, 2005, Journée d'étude de l'ATALA sur l'interface lexique-grammaire et les lexiques syntaxiques et sémantiques.

Sagot B., Danlos L., « Améliorer un lexique syntaxique à l'aide des tables du lexique-grammaire – Constructions impersonnelles et expressions verbales figées », *Cahiers du Cental*, 2007.

Thomasset F., Villemonte de la Clergerie E., « Comment obtenir plus des Méta-Grammaires », *Proceedings of TALN'05*, ATALA, Dourdan, France, June, 2005.

van Noord G., « Error Mining for Wide-Coverage Grammar Engineering », *Proc. of ACL 2004*, Barcelona, Spain, 2004.

Villemonte de la Clergerie E., « DyALog : a Tabular Logic Programming based environment for NLP », *Proceedings of 2nd International Workshop on Constraint Solving and Language Processing (CSLP'05)*, Barcelona, Spain, October, 2005.